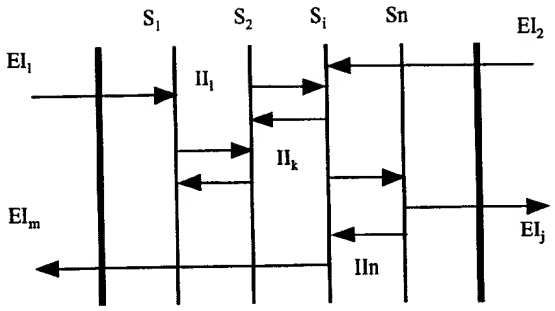
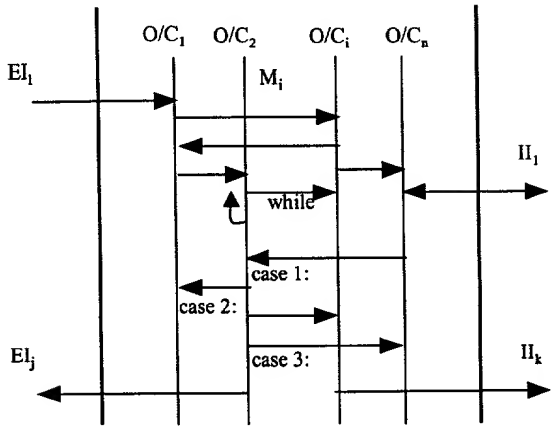


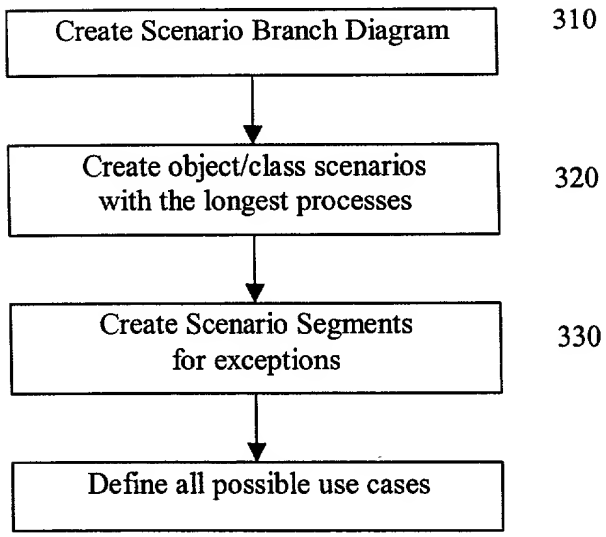
(Prior Art)

FIG. 1



(Prior Art)

FIG. 2



300

FIG. 3

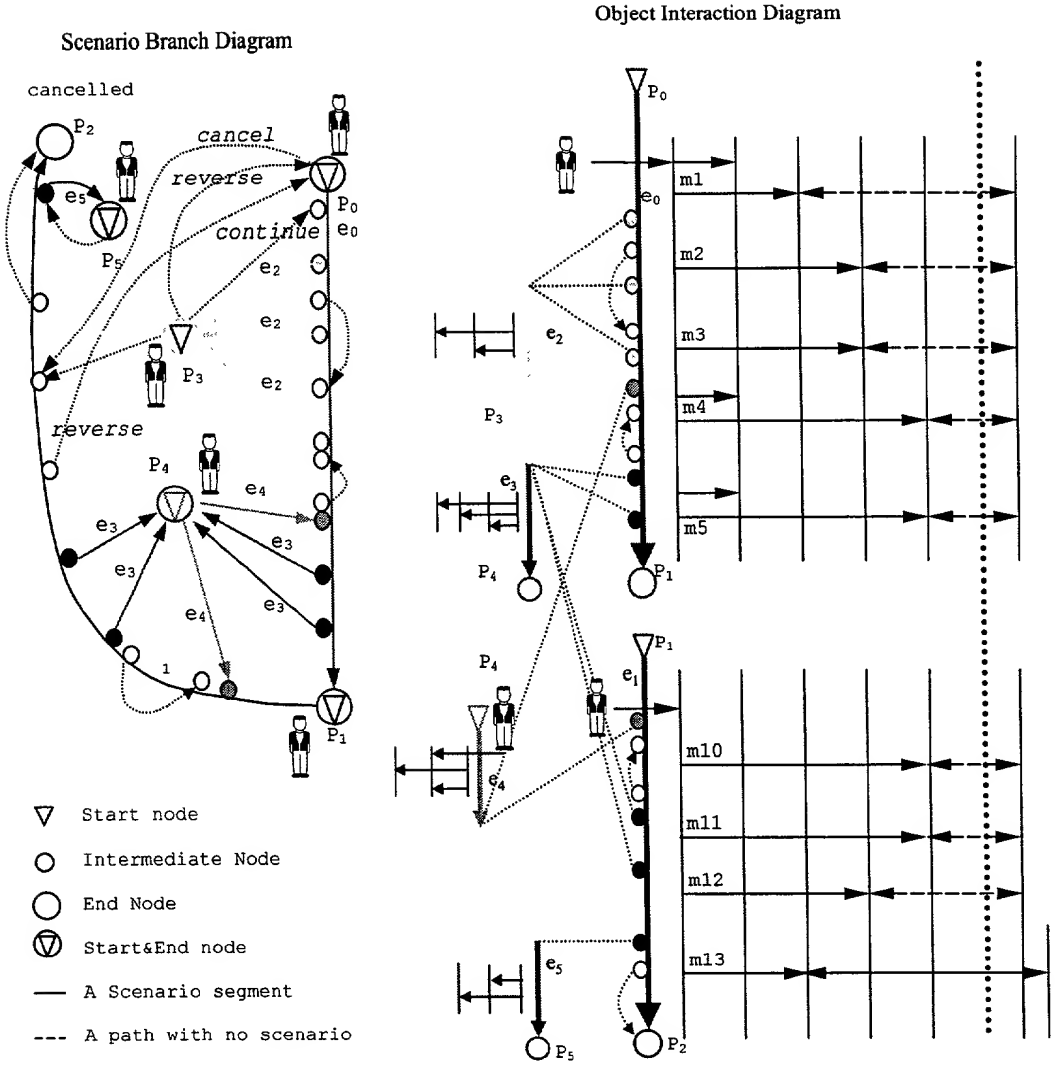
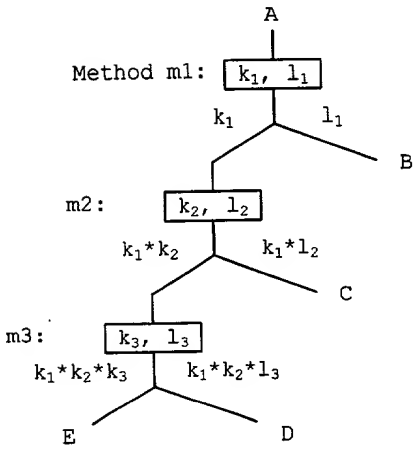

Scenario Branch Diagram
cancelled
P2
cancel
reverse
e5
P5
P0
e0
continue
e2
e2
e2
P3
reverse
P4
e4
e3
e3
e3
e3
e4
1
P1
Start node
Intermediate Node
End Node
Start&End node
A Scenario segment
A path with no scenario
Object Interaction Diagram
P0
e0
m1
m2
m3
m4
m5
e2
P3
e3
P4
P1
P1
P4
e1
m10
m11
m12
m13
e4
e5
P5
P2

FIG. 4

An example of use case set YPath execution complexity



If method i has k_i+l_i execution paths, where k_i is the number of paths to return "Success," and l_i is the number of paths to return "Failure", then

Use case A→E YPath = $k_1*k_2*k_3$
Use case A→B YPath = l_1
Use case A→C YPath = k_1*l_2
Use case A→D YPath = $k_1*k_2*l_3$

YPath is the maximum possible execution paths:
If k_1 includes paths of
"case U:..; case L:..; other..";
and k_2 also includes
"case U:… case L:… , other..";

After m_2, actual execution YPath=3 instead of 3*3=9;

FIG. 5

Field Name	Source: U	Source: L	Comment
r0	1	5	
r1	2	6	
r2	3	7	If (a>b) then...
r3	4	8	
r4	5	9	

resultCode=B; actionText=C;

Table basic semantic statement is:

```
if
((source=U)&(r0=1)&(r1=2)&(r2=3)&(r3=4
)&(r4=5)) {
    resultCode=B; actionText=C;
}
if
((source=L)&(r0=5)&(r1=6)&(r2=7)&(r3=8)
&(r4=9)) {
    resultCode=B; actionText=C;
}
```

FIG. 6

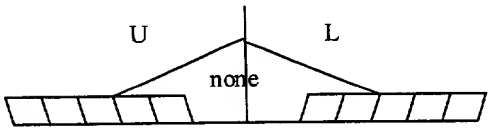


Fig 3.3: Basic logical complexity of Table 3.1

The first 5 cases are: if (ri<>..)..
6 th.: if ((r0=1)&(r1=2)&(r2=3)&(r3=4)&(r4=5))
Path=cyclomatic complexity in this table

FIG. 7

Field Name	r0	r1	r2	r3	r4	ResultCode	actionCode
Source: U	1	2	3	4	5	B	C
Source: L	5	6	7	8	9	B	C
Comment			if a.				

FIG. 8

Result Code	WC action	Source	Mail-address	Action Text
G_b_n	Full_S	U --- L	m1 --- m2	"..."
G_dup	No Action	U --- L	m1 --- m2	"duplicate" --- " "
G-dup_O	Fallout	U --- L	m1 --- m2	"duplicate" --- "Fallout"
S_dup	Action --- Add .. C	U --- L	m1 --- m2	Fallout
S_dup_O	Fallout	U --- L	m1 --- m2	Fallout

FIG. 9

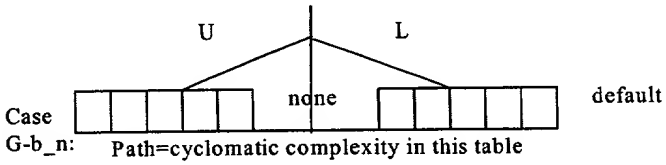


FIG. 10

Source	Result-Code	Mail-Address	Action-Text	WC-Action
U	G-h-n	M11	"..."	Full-S
	G_dup	M12	"duplicate"	
	G_dup_0	M13		
	S_dep	M14		
	S_dep_0q	M15		
L	G-h-n	M21		
	G_dup	M22		
	G_dup_0	M23		
	S_dep	M24		
	S_dep_0q	M25		

FIG. 11

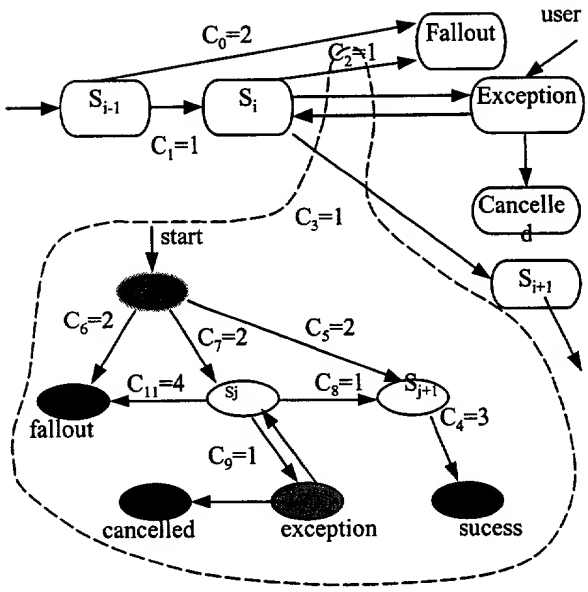


FIG. 12

s1	s2	S_{i+1}
working	working	(*no transition*)
	fallout1	Fallout
	fallout2	Fallout
	succeed	(*no transition*)
fallout	working	Fallout
	fallout1	Fallout
	fallout2	Fallout
	succeed	Fallout
Succeed	working	(*no transition*)
	fallout1	Fallout
	fallout2	Fallout
	succeed	Succeed

FIG. 13

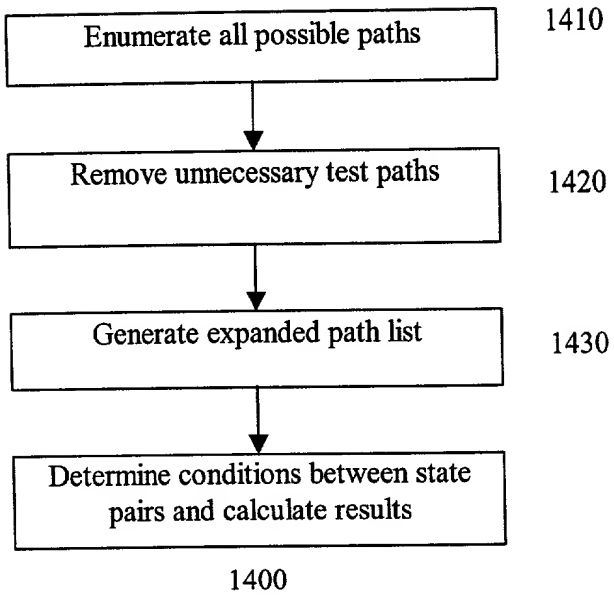
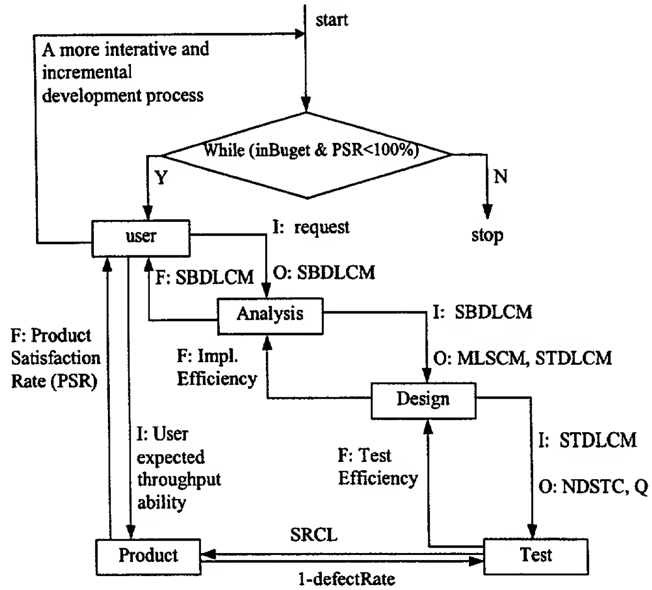


FIG.14

Data Type	In – Weight
Short, long, float, double, char, bool	0.5
String	1.0
Enum, union, sequence	1.5
Any	4.0
Struct, object, exception	Sum of in-weights of subfields
Array, vector, linklist	1.0 + in-weight of element type

FIG. 15



I: indicates input to the next stage;

O: indicates output of the stage;

F: indicates feedback to the previous stage.

SRCL: System release confidence level metric, presented in the attachment.

FIG 16